Submitted to the special edition of Physical Review Special Topic Accelerator and Beam

Efficient Numerical and Parallel Methods for Beam Dynamics Simulations

Jin Xu,[1,2] Brahim Mustapha,[1] and Misun Min[2]

[1]*Physics Division*
[2]*Mathematics and Computer Science Division*
Argonne National Laboratory
9700 S. Cass Ave., Argonne, IL 60439, USA

Abstract

This paper describes various numerical and parallel methods used at Argonne National Laboratory during the past five years to develop scalable software packages for beam dynamics simulations. It is based on the paper "Developing petascalable algorithms for beam dynamics simulations," which appeared in the proceedings of the first International Particle Accelerator Conference in 2010 [Jin Xu *et al*. Proceedings of IPAC10, Kyoto, Japan, May 23-28 (2010)]. Our focus here is on the standard particle-in-cell (PIC) method, direct Vlasov solvers, and scalable Poisson solvers. Among these methods the most challenging are scalable Poisson solvers, designed in three dimensions to account for space charge effects. Several approaches have been used to solve Poisson's equation efficiently in different situations. High-order numerical methods have been adopted to increase the accuracy. Domain decomposition methods have been used for parallelizing the solvers, and good scaling has been achieved. Preliminary results for the direct Vlasov solvers have been obtained for up to four dimensions. The parallel beam dynamics code PTRACK, which uses the PIC method to solve Poisson's equation, has been used as a workhorse for end-to-end simulations and large-scale design optimization for linear accelerators. We have successfully run the parallel methods on tens of thousands of processors of the IBM Blue Gene/P system at the Argonne Leadership Computing Facility, which provides an excellent environment for conducting large-scale beam dynamics simulations.

Key words: Poisson's Equation, Vlasov Equation, *hp*-FEM, Semi-Lagrangian Method, Discontinuous Galerkin Method, High Dimension

PACS subject classifications: 02.60.-x; 02.70.-c; 07.05.Tp; 52.65.-y;

1 Introduction

BDS plays a significant role in accelerator modeling, for both the design and the operation phases of an accelerator. Computational beam dynamics has become more powerful with the development of supercomputers capable of simulating more complex effects and phenomena in beam physics. As the computing era enters the petascale level, more accurate accelerator simulations can be conducted. To fully exploit this computing power, however, scientists need more efficient numerical and parallel methods. Charged particle beams are at the core of accelerator technology. Since a charged beam is a kind of plasma, most of the methods used for plasma simulations could be adopted and used for charged beam simulations. We divide the methods for simulating plasma into three categories: microscopic models, kinetic models, and fluid models.

In the *microscopic* model, each charged particle is described by six variables (x, y, z, v_x, v_y, v_z). Therefore, for N particles, there are 6N variables in total. Since every two particles have a mutual force interaction, this leads to a group of 6N equations each of which has 6N variables. Solving the beam dynamics equation in 6N dimensions exceeds the capability of current supercomputers for large N.

The *fluid* model is the simplest because it treats the plasma as a conducting fluid with electromagnetic forces exerted on it. This model is used for solving magnetohydrodynamics equations in 3D (x, y, z); MHD techniques solve for average quantities, such as density and charge. The fluid model has been used successfully to describe a continuous flow of fluid. Since charged beams are often accelerated and focused in bunches, the model is not suitable for charged beam simulations.

Between these two models is the *kinetic* model, which is used in most current beam dynamics simulations. This model obtains the charge density function by solving the Boltzmann or Vlasov equation in six dimensions (x, y, z, v_x, v_y, v_z). The Vlasov equation describes the evolution of a system of particles under the effects of self-consistent electromagnetic fields. The Vlasov equation can be solved in two ways. The most common way is the particle-in-cell (PIC) method, which uses the motion of the particles along the characteristics of the Vlasov equation using an Euler-Lagrange approach. The PIC method is fast and easy to implement; and, with the arrival of petascale computing, one-to-one simulation can be realized for low-density beams. But for more intensive beams, the PIC method still uses macroparticles, making it difficult to capture detailed beam structure. Furthermore, noise is associated with the finite number of particles in the simulations.

Another way to solve the kinetic model is to solve the Vlasov equation directly. This approach requires solving in $2 \times nD$ (n is the dimension of the physical space). Thus, for example, in order to simulate beams in three dimensions, the Vlasov equation must be solved in six dimensions. Clearly, petascale computing is required, and

petascale algorithms are critical to a successful solution.

During the past five years, Argonne National Laboratory has developed efficient numerical methods to meet these demands. This paper summarizes our efforts, focusing on PIC methods, direct Vlasov solvers, and Poisson solvers; their implementation in petascale software; and their application in beam dynamics simulations of linear accelerators on the IBM Blue Gene/P system at the Argonne Leadership Computing Facility. More specifically, the beam dynamics simulation (BDS) here refers to the simulation of high-intensity protons and heavy-ion beams in linear accelerators with space charge effects taken into consideration. Various electromagnetic elements for the acceleration, transport, deflection, and focusing of beams have been studied for simulating the beam from end to end (i.e., from the ion source where the beam is generated to the target where the accelerator ends). We note that this work is ongoing and that other methods are under development. We direct the interested reader to related papers for more detailed information [30,32–34].

The paper is organized as follows. The numerical methods are briefly explained in Section 2 and the parallel methods in Section 3; comparisons and discussions of these numerical methods are presented in Section 4. We draw our conclusions in Section 5.

2 Numerical Methods

In this section, we present the numerical methods used for BDS. Since the PIC method is the most widely used approach for simulating charged beams and plasma, it is presented in the first subsection. Next, direct Vlasov methods are presented. Their forms in 2D and 4D are introduced separately, and two schemes for time integration of the Vlasov equation are also presented. Since the space charge effects are needed for both the PIC and the direct Vlasov methods, the third subsection focuses on numerical methods for solving the Poisson equation.

2.1 Particle-in-Cell Method

The PIC method was already in use as early as the 1950s, and the method gained popularity for plasma simulation in the late 1950s and early 1960s through the work of Dawson [10], Hockney and Eastwood [19], Birdsall and Langdon [4], and others. In plasma physics applications, the method amounts to following the trajectories of charged particles in self-consistent electromagnetic (or electrostatic) fields computed on a fixed mesh. In this technique, particles are pushed by a particle mover, which integrates the equation of motion; Maxwell's equations determining the electric and magnetic fields are calculated by a field solver. The charges on the grid are obtained by distributing the charges of the charged particles on nearby grid nodes with suitable weights.

The PIC method is the most widely used approach in the kinetic model for BDS. It uses macroparticles to represent the charged particles in the beam, integrating the beam dynamics equations under external and internal forces. The external force usually comes from electromagnetic fields of accelerator components; the internal force comes from the space charge effect, which is accounted for by solving the Poisson equation on a fixed grid. Because of the relativistic effect, the beam dynamics equation should be solved in the appropriate coordinate system. Many software packages are available, such as RAYTRACE, PARMILA, IMPACT, and TRACK. Our BDS research is based on the beam dynamics code TRACK, developed in the Physics Division at Argonne National Laboratory over the past 10 years. The basic method is presented below.

The transport of a charged particle is described by the equation of motion:

$$\frac{d\vec{p}}{dt} = Q(\vec{E} + \vec{v} \times \vec{B}), \tag{1}$$

where $\vec{p}$ is the particle momentum and Q is its charge; $\vec{E} = \vec{E}_{ext} + \vec{E}_{int}$ and $\vec{B} = \vec{B}_{ext} + \vec{B}_{int}$ are the sums of external and internal electric and magnetic fields, respectively; and $\vec{v}$ is the particle velocity. TRACK uses six independent variables for tracking the phase-space coordinates of the particles, $(x, x' = dx/dz, y, y' = dy/dz, \beta = v/c, \phi)$, where $v = \mid \vec{v} \mid$ and ϕ is the particle phase shift with respect to a reference particle. Since the relativistic effect is being considered, the set of equations used for the step-by-step integration routine is

$$\frac{dx}{dz} = x', \frac{dy}{dz} = y', \frac{d\phi}{dz} = \frac{2\pi f_0 h}{\beta c} \tag{2}$$

$$\frac{dx'}{dz} = \chi \frac{Q}{A} \frac{h}{\beta\gamma} [\frac{h}{\beta c}(E_x - x'E_z) + x'y'B_x - (1 + x'^2)B_y + y'B_z] \tag{3}$$

$$\frac{dy'}{dz} = \chi \frac{Q}{A} \frac{h}{\beta\gamma} [\frac{h}{\beta c}(E_y - y'E_z) - x'y'B_x + (1 + y'^2)B_x - x'B_z] \tag{4}$$

$$\frac{d\beta}{dz} = \chi \frac{Q}{A} \frac{h}{\beta\gamma^3 c}(x'E_x + y'E_y + E_z), \tag{5}$$

where $\gamma = 1/\sqrt{1-\beta^2}; h = 1/\sqrt{1 + x'^2 + y'^2}; \chi = 1/m_a c^2; A$ is the mass number; m_a is the atomic mass unit; and $E_x, E_y, E_z, B_x, B_y, B_z$ are the components of electric and magnetic fields. TRACK uses the fourth-order Runge-Kutta method to integrate Equation (2); every time step is subdivided into four substeps. Currently, the external electromagnetic fields are computed by using commercial software packages, such as CST MWS, EMS, or ANSYS. (Several Poisson solvers also have been developed and are described in the following sections.) The code TRACK supports the following electromagnetic elements for acceleration, transport, and focusing of multicomponent ion beams:

- Any type of RF accelerating resonator with realistic 3D fields.
- Radio frequency quadrupoles.
- Solenoids with fringing fields.
- Bending magnets with fringing fields.
- Electrostatic and magnetic multipoles (quadrupoles, sextupoles, etc.) with fringing fields.
- Multiharmonic bunchers.
- Axial-symmetric electrostatic lenses.
- Entrance and exit of a high-voltage deck.
- Accelerating tubes with DC distributed voltage.
- Transverse beam steering elements.
- Stripping foils or films (for the Facility for Rare Isotope Beams, not general yet).
- Horizontal and vertical jaw slits for beam collimation.
- Static ion-optics devices with both electric and magnetic realistic three-dimensional fields.

More details can be found in [3,30]. The TRACK code has been used worldwide since 2000 and now is being used as a workhorse for design and optimization at Argonne. More details can be found at the website http://www.phy.anl.gov/atlas/TRACK/, including the executable, documentation, and examples, all of which can be downloaded without cost.

We point out that there exists another approach, different from ray-tracing codes, that uses the matrix formalism for the design and study of beam-optics systems, for example, TRANSPORT, TRACE3D, or GIOS. They are not included in this paper.

2.2 Direct Vlasov Method

The PIC method has some shortcomings, such as the noise associated with a finite number of macroparticles and the difficulty in describing the beam tail and eventual beam halo formation. Since PIC is a simplified method for solving the Vlasov equation, scientists are interested in solving the Vlasov equation directly. The direct Vlasov approach has the potential to compensate for the limitations of the PIC method, as it uses the distribution function—a more convenient approach for better description of beam structure including beam halo.

The direct Vlasov method solves the beam distribution function in the six-dimensional phase space (x, y, z, v_x, v_y, v_z). The evolution of the distribution function of particles $f(\vec{x}, \vec{v}, t)$ in the phase space $(\vec{x}, \vec{v}) \in \mathbf{R}^d \times \mathbf{R}^d$, with d =1,2,3 and time t, is given by the dimensionless Vlasov equation,

$$\frac{\partial f(\vec{x}, \vec{v}, t)}{\partial t} + \vec{v}(\vec{x}, t) \cdot \nabla_x f(\vec{x}, \vec{v}, t) + \vec{F}(\vec{x}, \vec{v}, t) \cdot \nabla_v f(\vec{x}, \vec{v}, t) = 0, \tag{6}$$

where the force field $\vec{F}(\vec{x},\vec{v},t)$ can be coupled to the distribution function f. Since beams are accelerated by electromagnetic fields, the Vlasov equation usually needs to be solved together with the Maxwell equations. If the external fields can be omitted and only the internal field needs to be considered, the Vlasov-Maxwell system becomes the Vlasov-Poisson system. The Poisson equation is solved with the macroscopic beam density ρ, which is obtained by integration of the charge distribution function.

$$\rho(\vec{x},\vec{v},t) = \int\limits_{\mathbf{R}^d} f(\vec{x},\vec{v},t)dv \tag{7}$$

The force field, which depends only on t and $\vec{x}$, is obtained by the gradient of the potential, which is determined by solving Poisson's equation,

$$\vec{F}(\vec{x},\vec{v},t) = \vec{E}(\vec{x},t), \qquad \vec{E}(\vec{x},t) = -\nabla_{\vec{x}}\phi(\vec{x},t), \qquad -\Delta_{\vec{x}}\phi(\vec{x},t) = \rho(\vec{x},t) - 1, \tag{8}$$

where $\vec{E}$ is the electric field and ϕ the electric potential.

The challenge in solving the Vlasov equation is its high dimensionality. To address this, we used for the time integration a time-splitting scheme proposed by Cheng and Knorr [6]. Their method transforms the Vlasov equation from 2D to two 1D equations. Our work in this area is preliminary but encouraging; it is presented briefly in Section 3. Details of these work can be found in [33,34]. In the following, the forms of the Vlasov equation in 2D and 4D are explained separately.

2.2.1 Vlasov Equation in 1P1V Phase Space

In 1P1V phase space, the normalized Vlasov equation can be written as follows [1].

$$\frac{\partial f(x,v,t)}{\partial t} + v(x,t)\frac{\partial f(x,v,t)}{\partial x} + E(x,t)\frac{\partial f(x,v,t)}{\partial v} = 0 \tag{9}$$

$$E(x,t) = -\frac{\partial \phi(x,t)}{\partial x}, -\Delta\phi(x,t) = \frac{\partial E(x,t)}{\partial x} = \rho(x,t) - 1 \tag{10}$$

$$\rho(x,t) = \int\limits_{-\infty}^{\infty} f(x,v,t)dv \tag{11}$$

The distribution function $f(x,v,t)$ is expanded on a structured quadrilateral grid, as shown on the right of Fig. 1. Modal bases, shown on the left of Fig. 1, have been used. A semi-Lagrangian method, explained in Section 2.2.3, has been used in the 2D plane. To increase the accuracy, we have adopted the algorithm proposed by

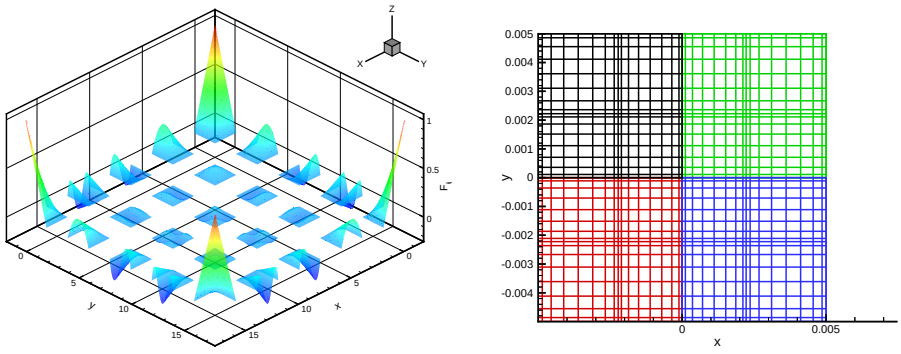

Fig. 1. Modal bases on quadrilateral (left) and quadrilateral mesh in a box domain (right).

Sonnendrücker in [27]. Each time step has been subdivided into three substeps: the first and the third substeps are in velocity space, and the second substep is in physical space. The detailed procedure follows.

. . . . . .

Do istep=1,nstep:
- Compute $j^n = q \int f^n(x^n, v^n) v^n dv$;
- Compute $E^{pred} = E^n - j^n \Delta t$ from Ampere's law;
- Do until $\mid E^{n+1} - E^{pred} \mid < \varepsilon$
  - Substep1: $v^{n+1/2} = v^{n+1} - E^{pred}(x^{n+1})\Delta t/2$
  - Substep2: $x^n = x^{n+1} - v^{n+1/2}\Delta t$;
  - Substep3: $v^n = v^{n+1/2} - E^n(x^n)\Delta t/2$;
  - Interpolate to compute charge density;
  - Solve Poisson's equation for E^{n+1};
  - Update new $E^{pred} = E^{n+1}$.
- Enddo

Enddo

. . . . . .

Since the semi-Lagrangian method relies heavily on the accuracy of the interpolation, using *hp*-FEM can easily achieve high order on each element by increasing the polynomial order. During each time-step iteration, Poisson's equation must be solved on a structured grid. The method is explained in Section 2.3.3. Although the accuracy can be high order in space, the current time integration scheme is only second order; we anticipate the adoption of high-order methods to increase the order of accuracy for time integration in the future. More details for solving the Vlasov equation in 2D can be found in [33].

2.2.2 Vlasov Equation in 2P2V Phase-Space

Next, we introduce our work on solving the Vlasov equation in higher dimensions, 2P2V. In beam dynamics, a simplified model was developed in 2P2V [14] as a paraxial model based on the following assumptions:

- The beam is in a steady state: all particle coordinates derivatives with respect to time vanish.
- The beam is sufficiently long so that longitudinal self-consistent forces can be neglected.
- The beam is propagating at constant velocity v_b along the propagation axis z.
- Electromagnetic self-forces are included.
- $\vec{p} = (p_x, p_y, p_z), p_z \sim p_b$ and $p_x, p_y \ll p_b$, where $p_b = \gamma m v_b$ is the beam momentum. It follows in particular that

$$\beta \approx \beta_b = v_b/c, \gamma \approx \gamma_b = (1 - \beta_b^2)^{-1/2}. \tag{12}$$

- The beam is thin: the transverse dimensions of the beam are small compared to the characteristic longitudinal dimension.

The paraxial model can be written as

$$\frac{\partial f}{\partial z} + \frac{\vec{v}}{v_b} \cdot \nabla_{\vec{x}} f + \frac{q}{\gamma_b m v_b}(-\frac{1}{\gamma_b^2}\nabla\Phi^s + \vec{E}^e + (\vec{v}, v_b)^T \times \vec{B}^e) \cdot \nabla_{\vec{v}} f = 0 \tag{13}$$

coupled with Poisson's equation,

$$-\Delta_{\vec{x}}\Phi^s = \frac{q}{\epsilon_0}\int_{R^2} f(z, \vec{x}, \vec{v}) d\vec{v}, \tag{14}$$

where Φ^s is the self-consistent electric potential due to space charge; $\vec{E}^e$ and $\vec{B}^e$ are the external electric and magnetic fields, respectively; and v_b is the reference beam velocity.

Two numerical methods have been applied in 2P2V simulations. One is a semi-Lagrangian method (SLM) on a structured grid in each 2D plane; the other is a discontinuous Galerkin (DG) method on an unstructured grid in each 2D plane. The methods are introduced in the following subsections. They both use the same time-splitting scheme proposed by Cheng and Knorr [6]. Each time step has been divided into three substeps: the first and third substeps are in physical space, and the second substep is in velocity space. The detailed procedure follows.

. . . . . .

Do istep=1,nstep:

- Substep 1: Perform a half time step shift in the (x,y) plane: $f^*(\vec{x}, \vec{v}) = f(t^n, \vec{x} - \vec{v}\Delta t/2, \vec{v})$
- Compute the electric field at time $t^{n+1/2}$ by substituting f^* in the Poisson's equation;
- Substep 2: Perform a full time step shift in the (v_x, v_y) plane: $f^{**}(\vec{x}, \vec{v}) = f^*(\vec{x}, \vec{v} - \vec{E}(t^{n+1/2}, \vec{x})\Delta t)$;
- Substep 3: Perform a second half time step shift in the (x,y) plane: $f(t^{n+1}, \vec{x}, \vec{v}) = f^{**}(\vec{x} - \vec{v}\Delta t/2, \vec{v})$;

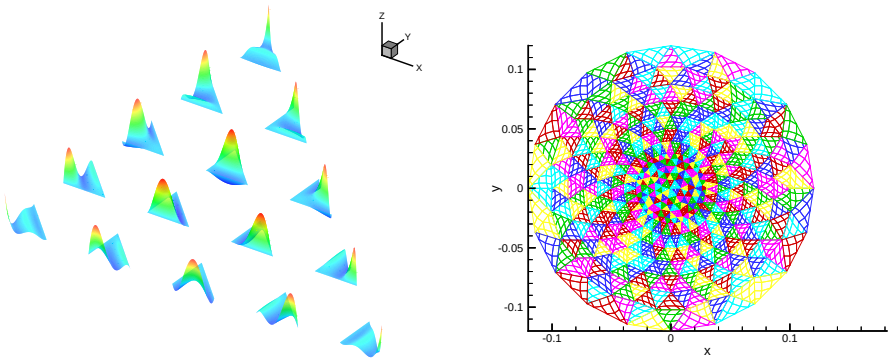

Fig. 2. Modal bases on a triangle (left) and triangular mesh in a circular domain (right).

Enddo

. . . . . .

On each two-dimensional space, the Vlasov equation transforms into a linear transport equation in the 2D plane. Two solvers have been developed for the 4D Vlasov equation. The first solver expands the distribution function $f(x, y, v_x, v_y, t)$ on a structured grid. The domain on each plane is a box. The bases and mesh used are shown in the left and right side of Fig. 1. The semi-Lagrangian method was used in 2P2V as in 1P1V. Instead of solving Poisson's equation and interpolating in 1D, however, the solution was done in 2D on structured quadrilaterals. Details can be found in [33].

The second solver expands the distribution function $f(x, y, v_x, v_y, t)$ on an unstructured grid. The domain on each plane is a circle. The bases and mesh been used are shown in the left and right side of Fig. 2. Using an unstructured grid can be more efficient because the beam usually has local concentration. Instead of using the semi-Lagrangian method, the discontinuous Galerkin method explained in Section 2.2.4 has been used. For the transport equation, researchers have successfully applied this method to the transient Maxwell and Euler equations in 2D and 3D. Since the split Vlasov equation on each 2D plane is a transport equation, we adopted a DG method to solve the Vlasov equation on each 2D plane. Furthermore, during the time-step iteration, Poisson's equation must be solved on an unstructured grid. A DG method called the internal penalty method was used to solve Poisson's equation on the same grid on the physical plane. Interested readers can find more details in [34].

2.2.3 Semi-Lagrangian Method

As shown on the left of Fig. 3, the semi-Lagrangian method consists of computing the distribution function at each grid point by following the characteristic curves backward and interpolating the distribution function at the previous time step. According to Liouville's theorem, the phase-space distribution function is constant along the trajectories of the system. Therefore, the interpolation at the previous time step

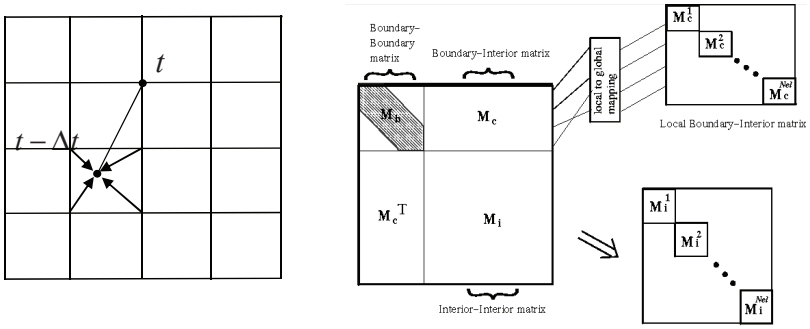

Fig. 3. Semi-Lagrangian method (left) and Shur complement method (right)

equals the function value at the present time step. In contrast to the Eulerian framework, the semi-Lagrangian scheme allows the use of large time-steps without losing stability. The limitations for stability are that trajectories should not cross and that particles should not "overtake" one another. Therefore, the choice of time-step size in the semi-Lagrangian scheme is limited only by numerical accuracy.

2.2.4 Discontinuous Galerkin Method

In 1973, Reed and Hill [23] introduced the first discontinuous Galerkin method for hyperbolic equations. Since that time there has been active development of DG methods for hyperbolic and nearly hyperbolic problems, resulting in a variety of methods. DG methods are locally conservative, stable, and high-order accurate methods. Originally, the DG method was realized with finite-difference and finite-volume methods. Later, it was extended to finite-element, *hp*-finite element, and spectral element methods. These make it easy to handle complex geometries, irregular meshes with hanging nodes, and approximations that have polynomials of different degrees in different elements. These properties have brought the DG method into many disciplines of scientific computing, such as computational fluid dynamics (especially for compressible flows), computational electromagnetics, computational plasma, semiconductor device simulation, chemical transport, and flow in porous media, as well as to a wide variety of problems such as Hamilton-Jacobi equations, elliptic problems, elasticity, and Korteweg-deVries equations. More details can be found in [7–9,18].

In the literature, the DG method has been used only in up to three dimensions. Since the Vlasov equation can involve higher dimensions, it brings new challenges to the DG method. Based on our successful experience in using a time-splitting scheme for solving the Vlasov equation directly [33], we have tried the DG method as a substitute for the semi-Lagrangian method in each substep, which is advanced in 2D phase spaces separately. Suppose that a 4D phase space Ω_K is composed of the tensor product of two 2D phase spaces, $\Omega^1_{K^1} \times \Omega^2_{K^2}$, and the total degrees of freedom in $\Omega^i_{K^i}$ is F^i. Then the total degrees of freedom in Ω_K is $F^1 \times F^2$.

Substituting t for z in Equation (13), we can write the time-splitting scheme

combined with the DG method as follows:

$$\frac{\partial f(\vec{x},\vec{v},t)}{\partial t}+\frac{1}{2}\nabla_{\vec{x}}\cdot[\vec{V}(x,y)f(\vec{x},\vec{v},t)]=0, \tag{15}$$

$$\frac{\partial f(\vec{x},\vec{v},t)}{\partial t}+\nabla_{\vec{v}}\cdot[\vec{U}(v_x,v_y)f(\vec{x},\vec{v},t)]=0, \tag{16}$$

$$\frac{\partial f(\vec{x},\vec{v},t)}{\partial t}+\frac{1}{2}\nabla_{\vec{x}}\cdot[\vec{V}(x,y)f(\vec{x},\vec{v},t)]=0. \tag{17}$$

The velocity in physical and velocity spaces is

$$\vec{V}(x,y)=\frac{\vec{v}}{v_b}=\frac{(v_x,v_y)}{v_b}, \tag{18}$$

$$\vec{U}(v_x,v_y)=\frac{q}{\gamma_b m v_b}[-\frac{1}{\gamma_b^2}\nabla\phi(x,y)+\vec{E}(x,y)]. \tag{19}$$

Another challenge in applying the DG method to solve the Vlasov equation is that the computer time increases as N^2, where N is the total degrees of freedom in physical and velocity spaces. Since in each subspace the transport equation is totally decoupled, it is well suited for large-scale parallel computing. Therefore, a highly scalable scheme could be developed, and the DG method matches this requirement, offering the added promise of high efficiency. More details can be found in [34].

2.3 Methods for Solving Poisson's Equation

Since the space charge effects are accounted for by solving Poisson's equation, several numerical methods have been adopted. Each has advantages in specific conditions, explained separately in the following. Poisson's equation is a standard second-order, elliptic partial differential equation, which has been studied extensively by the scientific computing world [16]. We briefly explain each method; details can be found in relevant publications. The most common methods for solving Poisson's equation, such as the finite-difference method and finite-volume method, are not included in this paper because they are available in most textbooks of scientific computing. The methods presented below are mostly numerical methods with high-order accuracy.

2.3.1 Fourier Spectral Method (FSM)

The Fourier spectral method is the standard method for solving Poisson's equation in a box region on a Cartesian coordinate system. More details can be found in

[5,15]. The potential has been expanded in Fourier series in all three directions. Either periodic or Dirichlet boundary conditions can be applied in all three directions.

$$\phi(x,y,z,t) = \sum_{m=-M/2}^{M/2-1} \sum_{p=-P/2}^{P/2-1} \sum_{n=-N/2}^{N/2-1} \hat{\phi}(m,p,n,t) e^{-i\alpha m x} e^{-i\beta p y} e^{-i\gamma n z} \tag{20}$$

Then Poisson's equation can be expressed as follows.

$$\begin{aligned} & f(x,y,z,t) = \Delta\phi(x,y,z,t) = \nabla^2\phi(x,y,z,t) \\ & = \sum_{m=-M/2}^{M/2-1} \sum_{p=-P/2}^{P/2-1} \sum_{n=-N/2}^{N/2-1} \hat{f}(m,p,n,t) e^{-i\alpha m x} e^{-i\beta p y} e^{-i\gamma n z} \\ & = -\sum_{m=-M/2}^{M/2-1} \sum_{p=-P/2}^{P/2-1} \sum_{n=-N/2}^{N/2-1} K(m,p,n) \hat{\phi}(m,p,n,t) e^{-i\alpha m x} e^{-i\beta p y} e^{-i\gamma n z} \end{aligned} \tag{21}$$

$$K(m,p,n) = \alpha^2 m^2 + \beta^2 p^2 + \gamma^2 n^2 \tag{22}$$

From this it is easy to see that $\hat{\phi}(m,p,n,t) = \hat{f}(m,p,n,t)/K(m,p,n)$. Parallel algorithms for this method are given in Section 3.4.1.

2.3.2 Fourier hp-Finite Element Method (Fhp-FEM)

Since most accelerating devices have round apertures, solving Poisson's equation in a cylindrical coordinate system (CYLCS) is more appropriate in this case. In CYLCS, Poisson's equation has the following form:

$$\nabla^2\phi(r,\theta,z) = \frac{\partial^2\phi}{\partial r^2} + \frac{1}{r}\frac{\partial\phi}{\partial r} + \frac{1}{r^2}\frac{\partial^2\phi}{\partial\theta^2} + \frac{\partial^2\phi}{\partial z^2} = -\frac{\rho}{\epsilon_0}, \tag{23}$$

where ϕ is the electrostatic potential, ρ is the charge density, and ϵ_0 is the permittivity of the vacuum. The 3D mesh is shown on the left of Fig. 4. The 2D mesh in the (r,θ) plane is shown on the right of Fig. 4. We use Gauss-Radau-Legendre quadrature points [20] in the first element close to the center to avoid $1/r$ singularity. In this case, there are four unequal elements in the radial direction. Each element has nine points, and each pair of adjacent elements share the boundary points. Periodic boundary conditions (BCs) have been applied in the longitudinal and circumferential directions, and a natural BC has been applied in the center of the cylinder. A Dirichlet zero BC has been applied at $r = r_0$, where r_0 is the radius of the cylinder.

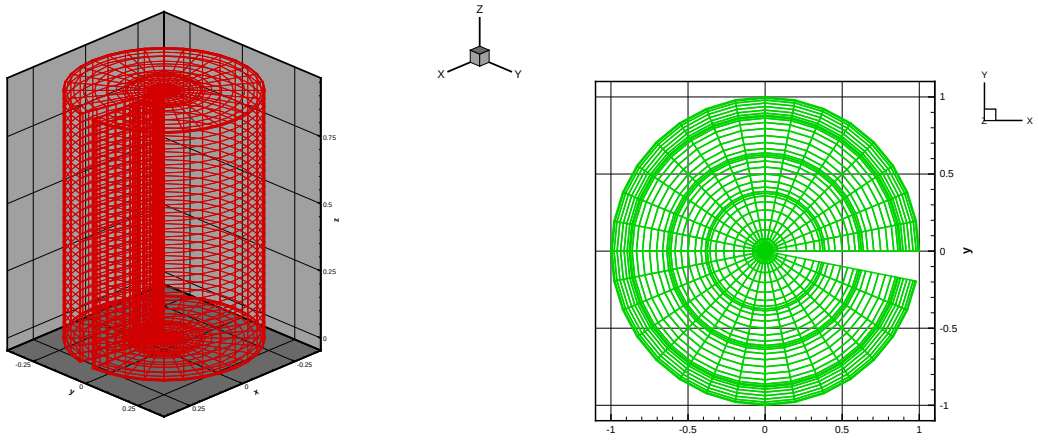

Fig. 4. 3D mesh in cylindrical coordinate system (left) and 2D mesh in the (r,θ) plane (right)

By transforming from the (r, θ, z) space to the (r, m, n) space through a Fourier transform, we obtain

$$\frac{1}{r}\frac{\partial}{\partial r}(r\frac{\partial\tilde{\phi}}{\partial r}) - \frac{m^2}{r^2}\tilde{\phi} - n^2\tilde{\phi} = -\frac{\tilde{\rho}}{\epsilon_0}. \tag{24}$$

This yields a linear system of equations

$$A \cdot \hat{\phi} = \hat{f}, \tag{25}$$

where A is a $(P+1)\times(P+1)$ matrix generated by the left-hand side terms, and $\hat{\phi}$, $\hat{f}$ are (P+1)-component vectors, where p,q=0,1,2,...,P. Equation (25) is solved directly since it is in one dimension. The size of A is not very large. Different Fourier modes m and n can be located on different processor to parallelize the solution. The procedure is explained in the next section, and more details can be found in [32].

2.3.3 hp Finite Element Method (hp-FEM)

The finite-element method has originated in the 1940s. It was developed from elastic and structural analysis for civil and aeronautical engineering. The accuracy of FEM can be increased by using a larger mesh, called *h*-FEM. It can also be improved by increasing the order of the bases, called *p*-FEM. The combination of these two approaches, called *hp*-FEM, is an area in numerical methods that has attracted many computational mathematicians [18,21,24,25].

The *hp*-FEM method has many advantages over popular, low-order methods in many applications [11,13,18,20,26]. The main advantages of *hp*-FEM are its flexibility in handling complex geometry and its high-order accuracy. Nearly all operations and data are local, including the derivative, interpolation, integration, solution of

Poisson's equation, and transformation between physical and coefficient spaces. It can use two types of bases: nodal or modal. Both modal and nodal *hp*-FEM have been successfully applied to solve the Poisson and Vlasov equations. In our work, the modal *hp*-FEM uses a structured mesh in 2D as shown on the left of Fig. 1, while the nodal *hp*-FEM uses the unstructured mesh in 2D geometry as shown on the right of Fig. 2.

With the Shur complement technique, the solution of a linear system $A \times x = b$ can be divided into two parts for boundary and internal modes, as shown on the right of Fig. 3. The boundary modes are solved iteratively by using a conjugate gradient method. The internal modes in each element can then be solved directly. The discrete system for the Poisson equation can be written as follows:

$$\begin{pmatrix} A_{bb} & C_{bi} \\ C_{bi}^T & A_{ii} \end{pmatrix} \begin{pmatrix} u_b \\ u_i \end{pmatrix} = \begin{pmatrix} f_b \\ f_i \end{pmatrix}$$

where b and i correspond to boundary and interior variables and u_b and u_i can be solved separately by the following equations.

$$(A_{bb} - C_{bi} A_{ii}^{-1} C_{bi}^T) u_b = f_b - C_{bi} A_{ii}^{-1} f_i \tag{26}$$

$$u_i = A_{ii}^{-1} (f_i - C_{bi}^T u_b) \tag{27}$$

More details can be found in [11,20].

Since the Vlasov solver on an unstructured grid is built on the DG framework, it is better to solve Poisson's equation with the DG method also. The method used in DG framework is called the interior penalty method [2,12,28]; more details can be found in [34].

3 Parallel Methods

In order to run on petascale supercomputers, the numerical methods explained above must be parallelized. As in the preceding section, the parallel methods are explained in three subsections. Since we have mainly used the IBM Blue Gene/P supercomputer at Argonne National Laboratory to test the parallel methods, we briefly introduce its structure and capability first.

3.1 BG/P Supercomputer

Argonne has two BG/P supercomputers: Surveyor and Intrepid. Surveyor has one rack (4,096 cores); Intrepid has 40 racks (163,840 cores). Each Blue Gene/P

rack contains 1,024 compute nodes. A compute node has four cores, each a 850 MHz PowerPC 450 processor with a dual floating-point unit ("double hummer"). The memory per node is 2 GB. There are 64 compute nodes per I/O node. An important feature of BG/P is the fast communication network. It has a 3D torus network that can achieve 5.1 GB/s per node. The peak performance of Intrepid is 557 teraflops, and it has a total of 80 TB of memory. The speed of each computing node is not very fast; but with the large number of computing nodes, BG/P supercomputers can achieve a faster speed than other types of supercomputers. Currently, several BG/P-type supercomputers are ranked in the Top500 list of fastest supercomputers on the world.

For our parallelization work, we have used the MPI library on the BG/P supercomputers at Argonne. The IBM xlC compiler has been used for the compilation, and both the BLAS and LAPACK libraries have been used for optimal numerical computation. All these libraries are available on different supercomputers, including clusters. Since our parallel methods are general, they can be used on almost any type of supercomputer. We note, however, that the recently emerged GPU-based systems have different software requirements; therefore, the implementation of our parallel methods needs modification for GPUs; but the parallel methods themselves remain the same as on a CPU-based system.

3.2 Parallel Method for PIC Software

Beam dynamics simulations involve two components: particle tracking (pusher) and space charge (SC) calculation (field solver). We have successfully parallelized our PIC-based beam dynamics code TRACK; the parallel version is called PTRACK [30]. The parallel algorithm for PTRACK is shown in Fig. 5.

At the beginning of the calculation, the internally generated or read-in initial particle distribution is equally distributed among all the processors. That is, each processor has only part of the total particles. But each processor has information about the full external fields for the beamline or accelerator element being simulated. The SC grid is also defined on all the processors. Before every tracking step the internal SC fields of the beam must be computed and combined with the external fields. The first step in the calculation of SC fields is the particle charge deposition on the nodes of the SC grid. This is done locally; that is, each processor deposits the charges of particles that are located on it. At the end of this step, every processor will have a partial SC distribution including only the charge of its particles. To calculate the SC distribution of the whole beam, we sum the partial SC distributions on the SC grid using the "MPI_Allreduce" routine of the MPI library [23]. To use domain decompositions of the Poisson solver, we subdivide the full SC grid into smaller-scale SC grids using 1D, 2D, and 3D space decompositions. Each processor will have a local SC grid containing only part of the SC data. FSM has been used for all

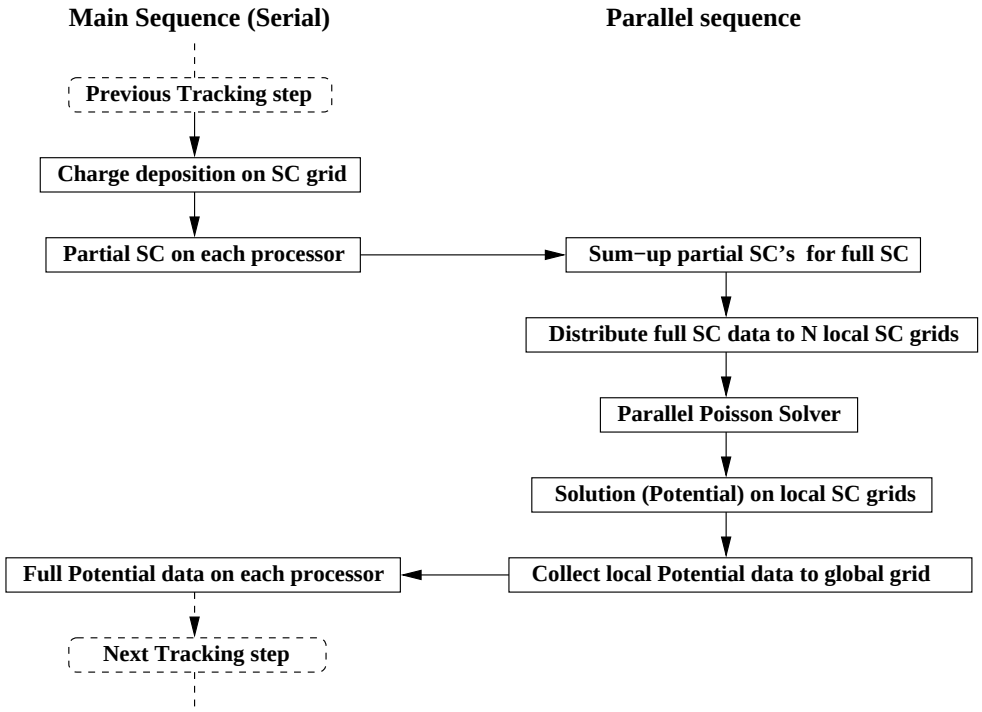

Fig. 5. Parallel algorithm of PTRACK

the decompositions. The fast Fourier transform (FFT) is a global transformation. Therefore, in order to perform the FFT in one direction, all data in that direction should be collected on one processor. After the FFT, they are distributed back to their original processors. After solving the Poisson equation, we have the solution in the form of potential data distributed on the local grids of each processor. In order to have all potentials on each processor, a second global communication using "MPI_Allreduce" brings the potential data from the local grids to the global grid to be ready for the tracking part of the calculation. The complete procedure is summarized in Fig. 5. PTRACK has been used mainly on the BG/P at Argonne. Documentation, executable and examples are available upon request.

The domain decomposition method has been used for the parallelization of other Poisson solvers, such as the Fourier *hp*-finite element method. More details can be found in [30].

3.3 Parallel Method for Direct Vlasov Solvers

For 1P1V, which is equivalent to a 2D simulation, 1D domain decomposition has been adopted in both the physical x and velocity v_x space. Two MPI communicators, *comx* and *comv*, have been generated for operations in different spaces. For 2P2V simulation using the structured grid, the parallel model performs 2D domain

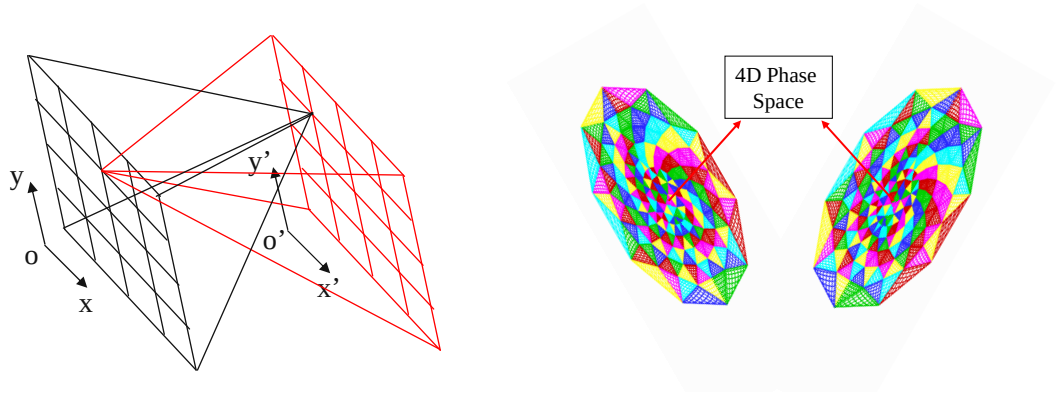

Fig. 6. Structured (left) and unstructured (right) grid in 4D domain by cross product of two 2D planes

decompositions in both the physical $(x,\ y)$ and velocity (v_x, v_y) planes; therefore, a 4D domain decomposition has been used, as shown on the left of Fig. 6. This makes it easy to use a large number of processors and is particularly helpful for direct solution of the Vlasov equation. Three splitting substeps are associated with the communicators. This approach makes it possible to solve the Vlasov equation in high dimensions. In these simulations, two more communicators, *comxy* and *comvxvy*, have been generated for operations in the different planes. The communicator *comxy* contains all processors with the same (v_x, v_y) location, and *comvxvy* contains all processors with the same (x, y) location. These two communicators are used for computing the beam statistics. When using the unstructured grid, only two communicators can be created, *comxy* and *comvxvy*, because it is impossible to distinguish x from y on the (x, y) plane and v_x from v_y on the (v_x, v_y) plane. Details in each case can be found in [33,34].

3.4 Parallel Methods for Poisson Solvers

Because of the global nature of Poisson's equation, involving the whole charge distribution to calculate an effective self-field created by all the particles in the beam, its parallelization is the most challenging part in developing scalable algorithms for beam dynamics simulations, especially when the grid is small and a large number of processors are used. In this section, we briefly describe the parallel algorithms developed with the domain decomposition method.

3.4.1 Parallel Method for FSM

Since FSM is the most popular method for solving Poisson's equation, more work has been devoted to this method. Three domain decomposition methods have been implemented, as shown in Fig. 7. With the appropriate model, it is easy to use tens of

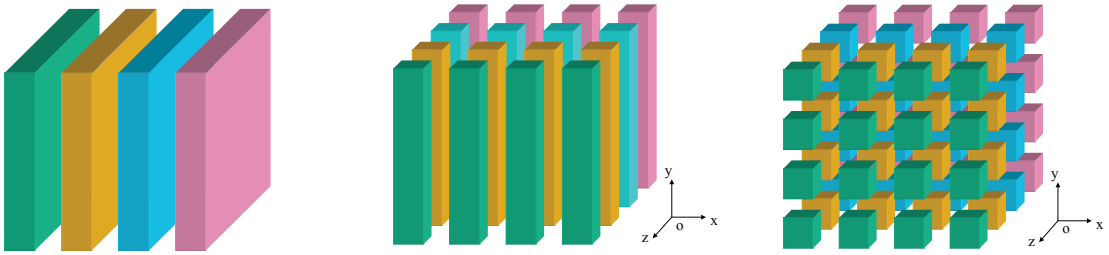

Fig. 7. Three parallel models for the Fourier spectral method

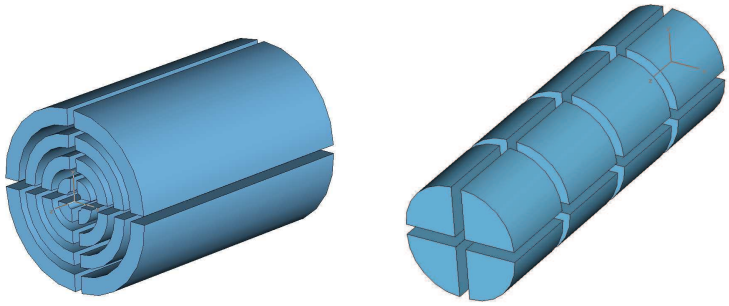

Fig. 8. Parallel models for the Fourier *hp*-finite element method

thousands of processors with a relatively small grid for the space charge calculation. For example, if the grid used for Poisson's equation is 32^3, then the maximum number of processors that can be used is 32 for the 1D decomposition model, $32^2 = 1,024$ for the 2D decomposition model, and $32^3 = 32,768$ for the 3D decomposition model. If the data is not located on one processor in any direction, a global $MPI\_Alltoall$ must be called to transfer the data to one processor. Another call is needed to transfer the data back to the original processors. Good scaling has been obtained with a relatively small grid for the space charge calculation; see [29–31].

3.4.2 Parallel Method for Fhp-FEM

For the Fhp-FEM in CYLCS, a 2D domain decomposition method has been developed, as shown in Fig. 8. The model on the right has the benefit of solving the 1D linear system $A \times x = b$ on each processor, as the equations are totally decoupled. The model on the left must solve the boundary modes first, then the internal modes in each element in the r direction. Since the element number in the r direction is not very large, this model is efficient and is being used for BDS. One can also develop a 3D domain decomposition approach, as was done for FSM, which can use a large number of processors. Details can be found in [33].

3.4.3 Parallel Method for hp-FEM

Parallel Poisson solvers using *hp*-FEM are achieved by using the Shur complement technique, shown on the right of Fig. 3. The boundary mode, Equation (26), has been solved by using an iterative conjugate gradient method, whereas the internal mode of each element, Equation (27), has been solved by the direct method. The mesh should be partitioned appropriately so that all processors have approximately the same numbers of elements to ensure load balance. In order to speed the conjugate gradient method, an efficient preconditioner can be used. For the multigrid technique, both the coarse and fine meshes use the same mesh partition. This makes it convenient to perform prolongation and restriction operations on each element. A good mesh partition is to minimize the modes located on the interfaces of different processors, as this will reduce the communication cost during global operations. More details can be found in [34].

4 Comparison and Discussion

In this section we compare the performance of the numerical methods presented in the paper.

4.1 PIC vs. the Direct Vlasov Method

The common point of PIC and the direct Vlasov method (DVM) is that they both need to solve the Poisson equation during each time step. PIC and DVM have many differences, however. PIC uses a Lagrangian approach, whereas DVM uses an Euler approach. Using a Lagrangian approach means that the macroparticles are traced during acceleration. Using an Euler approach means that the probability function in phase space is studied instead. As shown in Fig. 9, PIC uses macroparticles to represent a real beam, while DVM uses fixed grid points. The left image in Fig. 9 shows a beam bunch of 10^8 particles in 3D physical space. The middle and right images in Fig. 9 show that each point in the 3D physical space has a 3D velocity space associated with it. This makes DVM high dimensional, while in the PIC method a single macroparticle has a well-defined space and velocity coordinates. In the PIC method, macroparticles represent the same number of real beam particles, whereas in DVM the distribution function has different values on the grid points. Since PIC uses macroparticles, the statistics of the beam are obtained on particles, whereas for DVM they are obtained on grid points. Moreover, PIC is at most in 3D, whereas DVM can be in 6D.

To better understand their relation, we can shrink the velocity space in DVM to just one point and transform the DVM grid in physical space to exactly match the

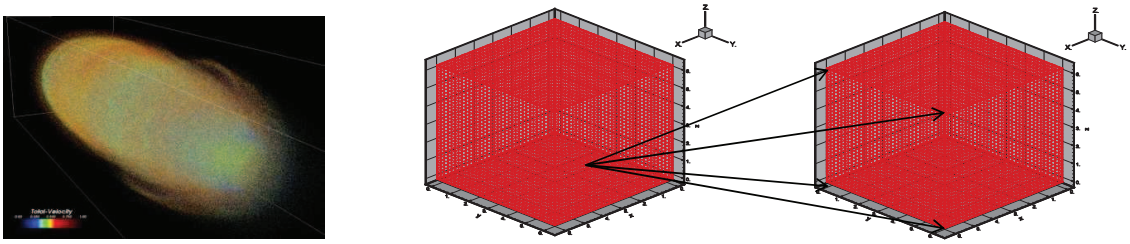

Fig. 9. Phase space for PIC (left) and Vlasov (right)

beam bunch. The DVM now translates to the PIC method: the DVM distribution function value at a grid point is equal to the particle numbers represented by the macroparticle located at that position.

Another difference between PIC and DVM is that DVM solves the transport equation in the velocity space, whereas PIC applies a force kick to represent the integration in the velocity space. This approach greatly simplifies the method and makes PIC easy to implement.

Since PIC uses more simplification and macroparticles to represent the same number of real particles, the method is difficult to use to capture the beam halo, where the particle number is less than the number represented by one macroparticle. DVM complements this method by using an Euler approach in which the distribution function in phase space is easy to use to represent the halo. On the other hand, large derivative and interpolation errors prohibit the use of DVM in real applications. More efficient numerical methods are needed that can simulate the real physics more accurately and efficiently. We plan to investigate this issue further.

4.2 Semi-Lagrangian vs. Discontinuous Galerkin Method

The semi-Lagrangian and discontinuous Galerkin methods are both efficient methods for time integration. Both methods have strong stability. While the DG method has stronger stability than does the continuous Galerkin method, the DG method has a strict limitation on the time-step size. SLM, on the other hand, is less limited and therefore, with the appropriate choice of the time-step size, can be more stable than the DG method. Since a beam is a concentrated bunch of particles, the distribution function can have sharp structures, making the errors of interpolation and derivation much larger than in usual plasma simulations. The greatest challenge comes from these large operator errors. In practice, SLM is easier to develop than the DG method. For parallelization, since the DG method has completely independent bases on each element, the communication is less except for the DG Poisson

solver. SLM, on the other hand, has to redistribute all backtracing points on different processors and thus is more complicated than the DG method. SLM also has a load-balancing problem because only a small portion of the phase space has a probability function larger than some value. Depending on the mesh partition, usually most backtracing points are located on some processors, an approach that wastes computing time. The DG method has a more balanced computing load because all processors have an equal amount of work to be done.

4.3 Poisson Solvers

We have presented several numerical methods for solving Poisson's equation. Among these, FSM is the most popular and efficient since the FFT algorithm can be used and since public FFT libraries are readily available. The drawback of FSM is the global characteristics of the Fourier transformation, which require that the data in one direction locate on one processor to complete the FFT. The large ratio of communication cost over the FFT cost makes the parallel efficiency low on a large number of processors. Since the total time is small compared to other time with PIC and direct Vlasov solvers, however, the overall parallel efficiency is still good. This is true of all three parallel methods developed for FSM, especially the one using 3D domain decomposition, which can be run efficiently on tens of thousands of processors.

Fhp-FEM is designed for CYLCS, which gives a more accurate solution for a cylindrical geometry because the boundary condition is more precisely defined on a cylinder wall. The number of processors used in Fhp-FEM usually is less than that of FSM, however.

Since hp-FEM is used in the radial direction, it, too, is limited in the number of processors that can be used.

High-order accuracy can be achieved by *hp*-FEM Poisson solvers when using high-order polynomials. Similar to FSM, however, there is some inherent restriction on scalability. Since the modes on interface between elements are solved globally through the conjugate gradient method, the parallel efficiency usually becomes worse when the number of processors increases. The parallel efficiency usually becomes better as the polynomial order increases for both the conjugate gradient and discontinuous Galerkin methods. The convergence of the conjugate gradient method is usually better than that of the DG method. Using an unstructured grid makes it easy to handle complex geometries.

The effectiveness of these Poisson solvers depends on the different configuration in which it is used. The solver should be chosen based on the geometry and boundary conditions.

4.4 Structured vs. Unstructured Meshes

Unstructured mesh is more appropriate for complex geometries; finer mesh can be generated in the central part of the beam. Both structured and unstructured meshes can apply a conjugate gradient or discontinuous Galerkin method, but the unstructured mesh is more complicated to develop. One significant difference between structured and unstructured meshes for the 2P2V Vlasov solvers is that we don't have access to the same statistical quantities. As explained in the previous section for parallel models for Vlasov solvers, with an unstructured mesh one cannot separate the x and y directions. There is no way to find all (x, v_x) points for a given (y, v_y) point. Similarly, there is no way to find all (y, v_y) points for a given (x, v_x) point. Therefore, the statistical quantities XX'_{rms} and YY'_{rms} cannot be calculated, whereas the statistical quantities X_{rms}, Y_{rms}, X'_{rms}, and Y'_{rms} can be obtained on both meshes. The definitions of these statistics can be found in [33].

5 Summary

We have briefly described numerical and parallel methods developed and used in our research for large-scale beam dynamics simulations during the past five years at Argonne National Laboratory. We hope these descriptions will be helpful to other researchers in accelerator simulations. Readers should consult the cited journal articles for more details.

The numerical methods have been discussed in three categories: PIC, direct Vlasov, and Poisson solvers. Parallel methods for all of them have also been described; these are designed to take advantage of petascale supercomputers. The parallel methods have been implemented in software packages and successfully run on tens of thousands of processors of the IBM Blue Gene/P at the Argonne Leadership Computing Facility. The parallel PIC-based beam dynamics code PTRACK has been used for large-scale beam dynamic optimization and real accelerator simulations. The parallel direct Vlasov solvers have been developed for both 1P1V and 2P2V simulations and have provided more detailed information on the physics, such as halo generation and filamentation in low dimension and with smooth initial distribution functions. Because of large derivation and interpolation errors, however, they are difficult to use in real accelerator simulations. More efficient numerical methods need to be developed to simulate complex beam dynamics more accurately.

Necessary for the success of PIC and direct Vlasov methods, several numerical methods for solving Poisson's equation in different situations have been presented and compared. They are critical to performing large-scale beam dynamics simulations.

Overall, these numerical and parallel methods serve as a basis for BDS, and

we expect that they will be applied to broader areas in the future. We note that the methods discussed in this paper represent only a selection of our research at Argonne. We also note that this work is ongoing. We expect to develop and use more efficient numerical methods and to optimize the current techniques, in order to improve beam dynamics simulations.

Acknowledgments

This paper is based on research conducted in the Physics and Mathematics and Computer Science divisions at Argonne National Laboratory during the past five years. Besides the co-authors listed, many other people have contributed to the work reported here, and we thank them for their contributions. We also thank Gail Pieper in the MCS Division at Argonne for her copyediting. This work was supported by the U.S. Department of Energy, Office of Nuclear Physics, under Contract No. DE-AC02-06CH11357.

References

[1] T.D. Arber and R.G.L. Vann, A critical comparison of Eulerian-grid-based Valsov solvers, *J. Sci. Comp.*, **180**, 339–357 (2002).

[2] D.N. Arnold, F. Brezzi, B. Cockburn, and L.D. Marini, Unified analysis of discontinuous Galerkin methods for elliptic problems, *SIAM J. Numer. Anal.,* **39**, no. 5, 1749–1779 (2002).

[3] V.N. Aseev, P.N. Ostroumov, E.S. Lessner, and B. Mustapha, TRACK: The new beam dynamics code. In *Proceedings of PAC-05 Conference,* Knoxville, Tennessee, May 16–20, 2005.

[4] C.K. Birdsall and A.B. Langdon, *Plasma Physics via Computer Simulation*, Inst. of Phys. Publishing, Bristol/Philadelphia (1991).

[5] C. Canuto, M.Y. Hussaini, A. Quarteroni, and T.A. Zang, *Spectral Methods in Fluid Mechanics*, Springer-Verlag, New York, 1987.

[6] C.Z. Cheng and G. Knorr, The integration of the Vlasov equation in configuration space, *J. Sci. Comp.*, **22**, 330–351 (1976).

[7] B. Cockburn, *Discontinuous Galerkin methods for convection-dominated problems.* In *High-Order Methods for Computational Physics*, edited by T. Barth and H. Deconink, Lecture Notes in Computational Science and Engineering, Vol. 9, Springer Verlag, Berlin, 1999, pp. 69–224.

[8] B. Cockburn, G. Karniadakis, and C.W. Shu, eds., The development of discontinuous Galerkin methods. In *Discontinuous Galerkin Methods: Theory, Computation and*

Applications, edited by B. Cockburn, G. E. Karniadakis, and C.-W. Shu, Lecture Notes in Computational Science and Engineering, Vol. 11, Springer Verlag, Berlin, 2000, pp. 3–50.

[9] B. Cockburn and C.W. Shu, Runge-Kutta discontinuous Galerkin methods for convection-dominated problems, J. Sci. Comp., **16,** 173–261 (2001).

[10] J.M. Dawson, *Particle Simulation of Plasma* Review of Modern Physics, **55,** 403 (1983).

[11] M.O. Deville, P.F. Fischer, and E.H. Mund, *High-Order Methods for Incompressible Fluid Flow,* Cambridge University Press, Cambridge, 2002.

[12] J. Douglas, Jr., and T. Dupont, *Interior Penalty Procedures for Elliptic and Parabolic Galerkin Methods*, Lecture Notes in Physics, Vol. 58, Springer Verlag, Berlin, 1976.

[13] M. Dubiner, Spectral methods on triangles and other domains, *J. Sci. Comp.*, **6**, 345–390 (1991).

[14] F. Filbet and E. Sonnendrücker, Modeling and numerical simulation of space charge dominated beams in the paraxial approximation, *Mathematical Models and Methods in Applied Sciences*, **16**, no. 5, 763–791 (2006).

[15] D. Gottlieb and S.A. Orszag, *Numerical Analysis of Spectral Methods: Theory and Applications,* SIAM-CMBS, 1977.

[16] L. Greengard and J. Lee, A direct adaptive Poisson solver of arbitrary order accuracy, *J. Comput. Phys.*, **125**, 415–424 (1996).

[17] J.S. Hesthaven and T. Warburton, Nodal high-order methods on unstructured grids, I: Time-domain solution of Maxwell's equations, *J. Comput. Phys.*, **181**, 186–221 (2002).

[18] J.S. Hesthaven and T. Warburton, *Nodal Discontinuous Galerkin Methods: Algorithms, Analysis and Applications,* Springer, New York, 2008.

[19] R.W. Hockney and J.W. Eastwood, *Computer simulation using particles* Adam Hilger, New York, 1988.

[20] G.E. Karniadakis and S.J. Sherwin, *Spectral/*hp *Element Methods for CFD,* Oxford University Press, London, 1999.

[21] T. Koornwinder, Two-variable analogues of the classical orthogonal polynomials. In *Theory and Applications of Special Functions*, edited by M. Ismail and E. Koelink, Academic Press, San Diego, 1975.

[22] A.T. Patera, A spectral element method for fluid dynamics: Laminar flow in a channel expansion, *J. Comput. Phys.*, **54,** 468–488 (1984).

[23] W.H. Reed and T.R. Hill, *Triangular mesh methods for the neutron transport equation*, Tech. Report LA-UR-73-479, Los Alamos Scientific Laboratory, Los Alamos, NM, 1973.

[24] G. Strang and G.J. Fix, *An Analysis of the Finite Element Method,* Wellesley-Cambridge Press, Wellesley, 2008.

[25] B. Szabó and I. Babuška, *Finite Element Analysis,* Wiley, New York, 1991.

[26] J. Shen and T. Tang, *Spectral and High-Order Methods with Applications,* Science Press of China, 2006.

[27] E. Sonnendrücker, F. Filbet, A. Friedman, E. Oudet, and J.-L. Vay, Vlasov simulations of beams with a moving grid, *Comput. Phys. Comm.*, **164**, 2390–395 (2004).

[28] M.F. Wheeler, An elliptic collocation-finite element method with interior penalties, *SIAM J. Numer. Anal.*, **15**, 152–161 (1978).

[29] J. Xu, Benchmarks on tera-scalable models for DNS of channel turbulent flow, *Parallel Computing.* **33/12**, 780-794 (2007).

[30] J. Xu, B. Mustapha, V.N. Aseev, and P.N. Ostroumov, Parallelization of a beam dynamics code and first large scale radio frequency quadrupole simulations, *Physics Review Special Topic–Accelerator and Beams,* **10**, 014201 (2007).

[31] J. Xu, B. Mustapha, V. N. Aseev, and P.N. Ostroumov, Simulations of high-intensity beams using BG/P supercomputer at ANL. In *Proceedings of HB 2008,* Nashville, Tennessee, Aug. 25–29 (2008).

[32] J. Xu and P.N. Ostroumov, Parallel 3D Poisson solver for space charge calculation in cylinder coordinate system, *Computer Physics Communications,* **178,** 290–300 (2008).

[33] J. Xu, B. Mustapha, P.N. Ostroumov, and J. Nolen, Solving 2D2V Vlasov equation with high order spectral element method, *Communications in Computational Physics,* **8**, 159–184 (2010).

[34] J. Xu, P.N. Ostroumov, J. Nolen, and K.J. Kim, Scalable direct Vlasov solver with discontinuous Galerkin method on unstructured mesh, *SIAM J. Scientific Computing* **32**, no. 6, 3476-3494 (2010).